Third year

Communications

Data structures

(Second term 2011-2012)

(15)

(Final Revision - Part II)

Price: "7" L.E

Tel. no.: 010-03654726

Cairo University
Dept. of Electronics & Communications Engineering
Third Year
Maximum grade is 70 points

Faculty of
Co
Time:

أجب على الأسئلة الآتية
يرجى التأكد من وجود ٦ أوراق بالكراسة مقفلة مدعمه ببسماء في نهاية الكراسة
في حالة عدم كفاية المساحة المخصصة لإجابة أحد الأسئلة اكمل الأجابة في الصفحة المقابلة له أو في المساحة المخصصة نهاية الكراسة

Question #1 (Total 12 points)

Define and implement the class `sq_matrix` which represents a square matrix of doubles. The dimension of the matrix is less than 100. The class supports the following operations: (4 points for proper class declaration)

- (2 points) `square_matrix(unsigned int dimn)` : it sets the dimension of the matrix to `dimn` and the elements are initialized to zeros. It must check that `dimn` is less than 100.
- (3 points) `Add(const matrix &B)` : which adds matrix `B` to `current` matrix. The operation is only if the two `matrices are of the same dimension.`
- (3 points) `Copy(const matrix &B)` : which copies matrix `B` to `current` matrix overriding dimension and stored values .

١

Question #2 (Total 20 points)

Using the standard singly linked-list storing items of `ListElemType`, add the following member functions:

- `Split(ListElemType pivot, List &L2)` which splits the list into two lists, all elements smaller than pivot remain in the current list and all elements greater than or equal to pivot move to list L2.
- `Sort( )`: This function sorts the current list using the selection sort algorithm.

Write the definition of the class (4 points) and write the code of the above two member functions (8 points each)

2

Question#3 (Total 14 points)

a) (6 points) Write the code for binary search algorithm that searches a sorted array a [ ] of integers for the el
equal to key. The array size is passed as parameter n.

b) (4 points)

c) (4 po

n searching for 24.

3

Question#4 (Total 16 points)

a) (8 points) Write an external function `Boolean Identical(const Queue &Q1, const Queue &Q2)` that determines if the two queues are identical. Preconditions: Q1 and Q2 have been initialized. Postconditions: Queues are unchanged and function return value is true if the two queues are identical, false otherwise.

b) (8 points) Write an external function `int Replace(Stack &S, StackElement oldValue, StackElement newValue)` that uses member functions of the stack ADT to replace all occurrences of oldValue in the stack S with newValue. Restore the rest of the stack unchanged.

4

Question#5 (Total 18 points)

Consider the implementation of the chained hash tables, with the standard operations of lookup, deleteKey, and
The entries in the table is kept in an array with maximum size `MAX_TABLE_SIZE`, however, the actual table
can be smaller than `MAX_TABLE_SIZE` and is kept in a private variable called `tableSize` that is passed as
parameter to the constructor, i.e. the constructor of the Table class will look as follows:
`Table::Table(int t_size.` For simplicity you may disregard the usage of templates for this question.

i- (6 points) Write the code for the new constructor function which should assure that
`t_size <= MAX_TABLE_SIZE`. If `t_size` is equal to zero, the table size becomes `MAX_TABEL_SIZE`.

ii- (1
fol
`ble &t1)` that copies the
`t1`
are
`rrent` table is erased and
its
in

Cairo University
Dept. of Electronics & Communications Engineering
Third Year
Maximum grade is 70 points

Faculty of Engineering
May 2010
Computers 3B
Time: 2 hours

أجب على الأسئلة الآتية

يرجى التأكد من وجود ٦ أوراق بالكراسة مشتملة صفحة بيضاء فى نهاية الكراسة

فى حالة عدم كفاية المساحة المخصصة لإجابة أحد الأسئلة، أكمل الأجابة فى الصفحة المقابلة له أو فى المساحة المخصصة نهاية الكراسة

Question #1 (Total 15 points)

Define and implement the class `Polynomial` which represents polynomial of the form $P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0$. The degree of the polynomial `n` is less than 100. The class supports the following operations: (4 points for proper class declaration)

- (2 points) `Polynomial (unsigned int deg)`: it sets the degree of the polynomial to `deg` and all the coefficients $\{a_i\}_{i=0}^{deg}$ are initialized to zeros. It must check that `deg` is less than 100.
- (3 points) `Polynomial (unsigned int deg, double a[])`: it sets the degree of the polynomial to `deg` and all the coefficients $\{a_i\}_{i=0}^{deg}$ are initialized with the values in the array `a[]`. It must check that `deg` is less than 100.
- (5 points) `Polynomial* Add(const Polynomial B)`: which adds polynomial `B` to the `current` polynomial and returns a pointer to the resulting polynomial. The degree of the resulting polynomial is the larger of the two polynomials' degrees.

Question #2 (Total 18 points)

Define a single-linked circular list (CircList) ADT supporting the standards list operations of `First`, `Next`, `Insert`. In addition the list supports the operations `void Merge(const CircList &clist)` which merges `clist` with current list and the operation `ReversePrint` which prints the elements in the list in reverse order without using any additional structures. Write the definition of the class and the functions `Insert`, `Merge`, and `ReversePrint`.

[illegible] (Total 12 points)

b) (6 points) Write an external function `int Replace(Queue &Q, QueueElemType item, QueueElemType newValue)` that uses member functions of the queue ADT to replace all occurrences of `item` with `newValue` leaving the rest of the queue unchanged. The functions returns the number of queue entries that were replaced.

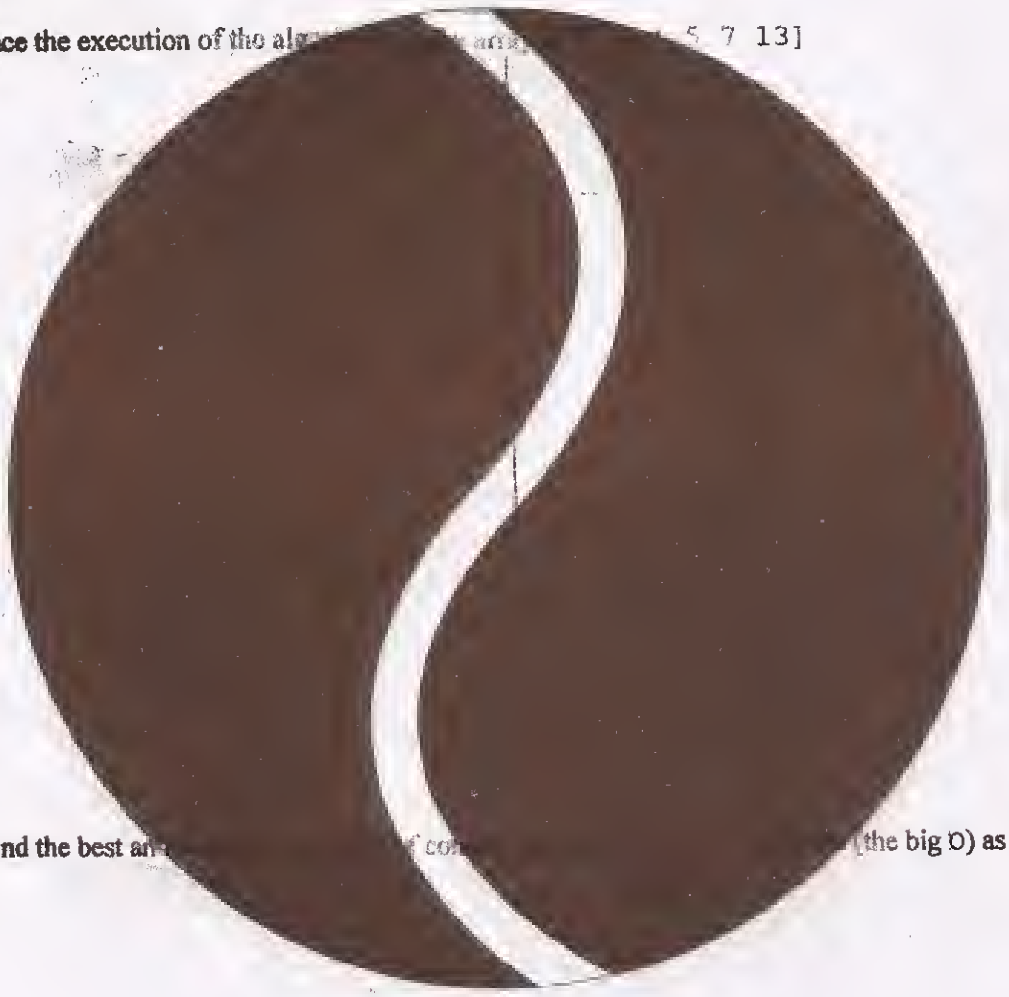
b) (6 points) Show how the following [illegible] expression [illegible] What is the final result? 5 3 * 8 – 4 + 3 2 / – +

Question#4 (Total 14 points)

The following is an algorithm known as InsertSort that sorts array A in ascending order.

```
void insertionSort(int a[], int n)
{
      int x; // item in the array
      int i, j;
      for (i=1; i < n; i++){
            x = a[i];
            j = i - 1;
            while (j >=0 && a[j] > x){
                  a[j+1] = a[j];
                  j--;
            } // end while
            a[j+1] = x;
      } // end for
}
```

a) [8 points] Trace the execution of the algorithm on the array [3 5 7 13]



b) [6 points] Find the best and worst case complexity (the big O) as function of n (array size).

Best case:

Worst case:

a) [10 points] Classify the following statements as T or F, with brief explanation why.

i) Binary search is an O(n log n) algorithm.

ii) Binary search is always faster than sequential search.

iii) When hashing is used increase size of hash table array always reduces the number of collisions.

iv) If we use chaining in a hashing scheme we don't have to worry about collision resolution.

v) For population size of 50, a table [illegible] prime [illegible] will always provide better collision performance if table [illegible]

b) [10 points] Consider the keys 6 47 87 9 126 [illegible] 567 566 620 735. Show how these values will be stored in a hash table:

i) Using linear probing with a hash table of size 20.

ii) Using chained hashing with a table of size 10.

Then fill the provided table indicating the number of [illegible] key.

Linear Probing

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	

[illegible]ng

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

#of comparisons

Key	Linear Probing	Chained Hashing
87		
566		
153		

10

...onics & Communications Engineering

4

...e is 70 points

أجب على الأسئلة الآتية

يرجى التأكد من وجود ٦ أوراق بالكراسة مشتملة صفحة بيضاء فى نهاية الكراسة

فى حالة عدم كفاية المساحة المخصصة لإجابة أحد الأسئلة، أكمل الإجابة فى الصفحة المقابلة له أو فى ...

المخصصة نهاية الكراسة

...1 (20 points)

...briefly what is meant by the following terms and their role in Object-Oriented Design and Programming:

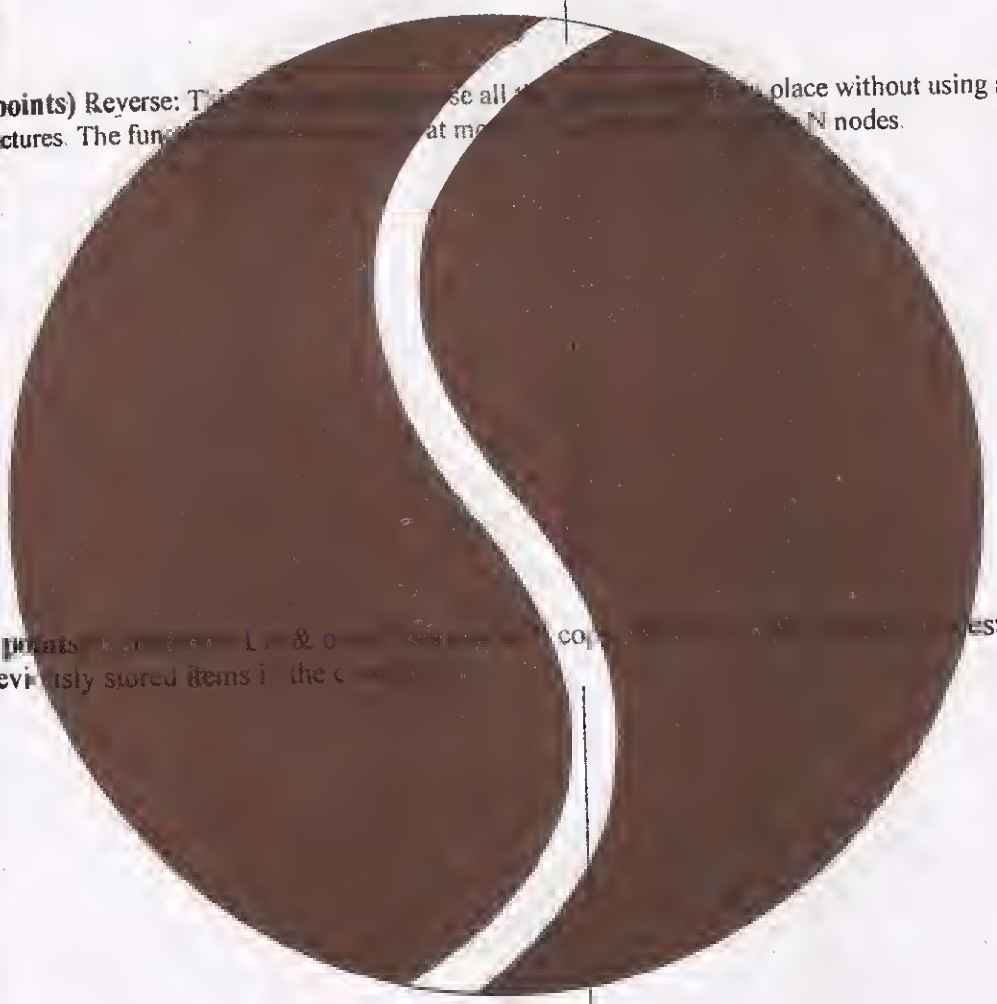
Abstraction

...efine an abstra... ...plex numbers. Each ...lement of the vect... ...mpl... ...he Complex ADT ...upports the operation... ...ful... ...he ComplexVector class ...ntains upto 100 comple... ...ris ... ...n in the table below. Write ...he definition of the Complex ... ...plex... ...tions. You need to select the ...roper return types for all member fu... ...on't n... ...e the Complex class ADT.

...mplexVector(int Size)	The class constructor that sets the size of the complex vector to Size and initialized all elements of the vector to (0+j0).
...Elem(int i, const Complex	Sets the value of element i of the vector to c. It should check that i is less than size of the vector.
...dd(const ComplexVector &x)	Adds the vector x to the *current* object. The function should first assert that the two vectors have the same size, if so the addition is performed.
...otProduct(const ...plexVector &x)	Calculates the dot product of the vector x and the *current* object. The function should first assert that the two vectors have the same size, if so the addition is performed.

standard single-linked list ADT storing items of ListElemType. Write code for the following member

nts) Interchange: This function will consider each two adjacent nodes together and exchange the data in So, starting with the head as node 0, the contents of nodes 0 and 1 will be exchanged, nodes 2 and 3 will be nged and so on. If the list contains an odd number of nodes, the contents of the last node will not be rged with any other node.



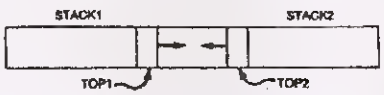
6 points) Reverse: T se all place without using any additional ructures. The fun at m N nodes.

(8 p co estroying all previ tsly stored items i the c

12

(16 points)

ts) Two stacks are implemented in one array of length 100 as shown. The tops of the stacks are pointed to top2 and grow toward each other. Write a procedure int ataType v, int s) that pushes the contents of v onto the stack indicated by s. value of 1 or 2. If the stack to which v is to be added is full, then push ld it, but should return -1, if push succeeds it returns s. At initialization d top2 = 100. Provide the condition for a full stack.



points) element of a queue, ng other el alue indicating er the oper

Canceled & solved before

20 points)

implementation of the chained hash tables with keys of type KeyDataType and associated data of ..., supporting the standard operations lookup, deleteKey, and insert. The entries in the table is kept in an ...aximum size *MAX_TABLE_SIZE*, however, the actual table size can be smaller than *MAX_TABLE_SIZE* ...n a private variable called *tableSize* that is passed as parameter to the constructor, i.e. the constructor of ...ss will look as follows: *Table::Table(int t_size)*. For simplicity you may ignore templates usage for this

...nts) Write the code for the new constructor function. Make sure that *t_size* <= *MAX_TABLE_SIZE*.

...oints) Add a member funct... follow... *...onst Table &t1)* that merges the current table *t* ... the table *t1*. Keep in... ...bles in... ...lues of *tableSize*.

...r key lkey. If and ...ist (so if the node ... so on). If the node is ...

19

Final 2009 :

Question 1 :

```
Const int maxsize = 100;
class sq_matrix {
 private :
    double
    int
 public :
    sq_
Void Add
Void Copy (Const
                 &B);
};
```

```
sq_matrix :: sq_matrix
        (unsigned int dimn)
{ assert (dimn < maxsize);
  dim = dimn;
  for (int i=0; i<maxsize; i++)
    for (int j=0; j<maxsize; j++)
      A[i][j] = 0;
}
```

```
sq_matrix :: Add
        sq_matrix & B)
{
  B.dim);
  im; i++)
  dim; j++)
    B.A[i][j]
}
```

```
:: Copy
  sq_matrix & B)
{ dim = B.dim;
  for (int i=0; i<dim; i++)
    for (int j=0; j<dim; j++)
      A[i][j] = B.A[i][j];
}
```

Final 2009 :

Question 1 :

```
Const int maxsize = 100;
class sq_matrix {
 private :
    double
    int
 public :
    sq_
Void Add
Void Copy (cons
                 &B);
};
sq_matrix :: sq_matrix
        (unsigned int dimn)
{ assert (dimn < maxsize);
  dim = dimn;
  for (int i=0; i<maxsize; i++)
   for (int j=0; j<maxsize; j++)
     A[i][j] = 0;
}
_matrix :: Add
        sq_matrix & B)
        B.dim);
        im; i++)
        dim; j++)
        B.A[i][j]
:: Copy
sq_matrix & B)
{ dim = B.dim;
  for (int i=0; i<dim; i++)
   for (int j=0; j<dim; j++)
     A[i][j] = B.A[i][j];
}
```

Question 2:

* class definition:

```
typedef int listElemType;

class list {
  private:
    struct node;
    typedef node*
    struct n
    listElemType
    lin    next; };
    lin  head; link cu
  public:
    list ();
    bool find (listE
    next(
    void insert(
    void Split (list
      pivot, list & L2);
    void Sort ();
};
```

* void list :: split

```
(listElemType elem,
  list & L2)
{
  link del;
  ile (head &&
    ->elem >= pivot)
      t (head->elem);
      ;
      h    ->next;
    del
  }
  h    d;
  & pred->next;
  ->next->elem
    >= pivot)
{ L2.insert(pred->next->
    elem);
  del = pred->next;
  pred->next = del
    ->next;
  delete del;
}
else
  pred = pred->next;
} }
```

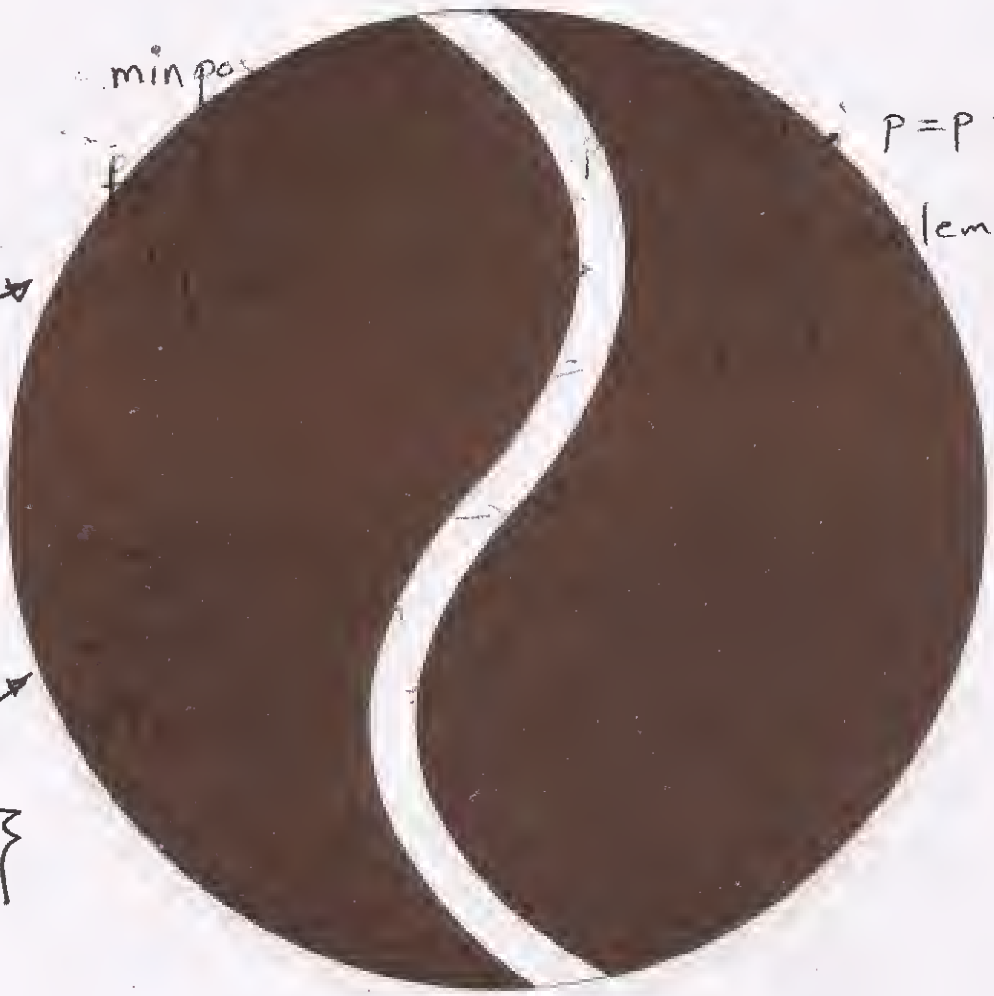
```
Void list::sort ()
{
  link minpos, p; ListElemType temp;
  for (link ptr=head; ptr != ptr->next;
       ptr = ptr->next)
  {
    minpos
    p=p->next)
    lem)
  }
}
```

op for

rems, list

arch for min

op in with first elem



Question 3:

- a & b : See notes
- c) first step:

first = 0 , last = 6 ,

$$mid = \frac{0+6}{2} = 3$$

a[3] < key

17 24

first = mid+1 = 4

Second step:

- first = 4, last = 6, mid = $\frac{4+6}{2} = 5$
- a[5] < key

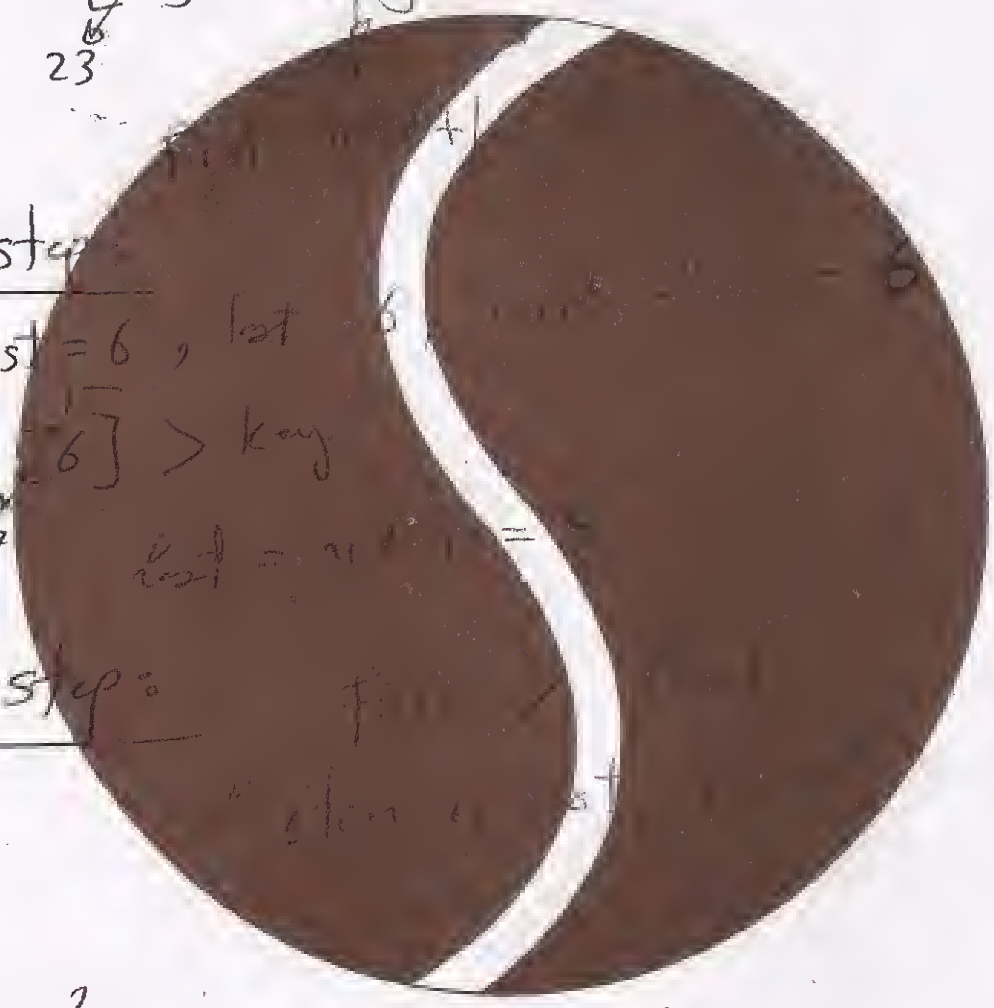
23

Third step:

- first = 6, last
- a[6] > key

27

Fourth step:



Question 4:

[a] bool identical (const Queue & Q1, const Queue & Q2)

{ Queue Q3, Q4; bool flag = true;
QueueElemType x, y;

```
bool temp1 = Q1.isEmpty();
bool temp2 = Q2.isEmpty();

while (!temp1 && !temp2)
{
    x = Q1.dequeue();   Q3.enqueue(x);
    y = Q2.dequeue();   Q4.enqueue(y);
}

while (
  Q1.en
while (!Q
  Q2.enqueue(Q

if ((!temp1 && !temp2) || temp1 || temp2)
    flag = false;

return flag;
}
```

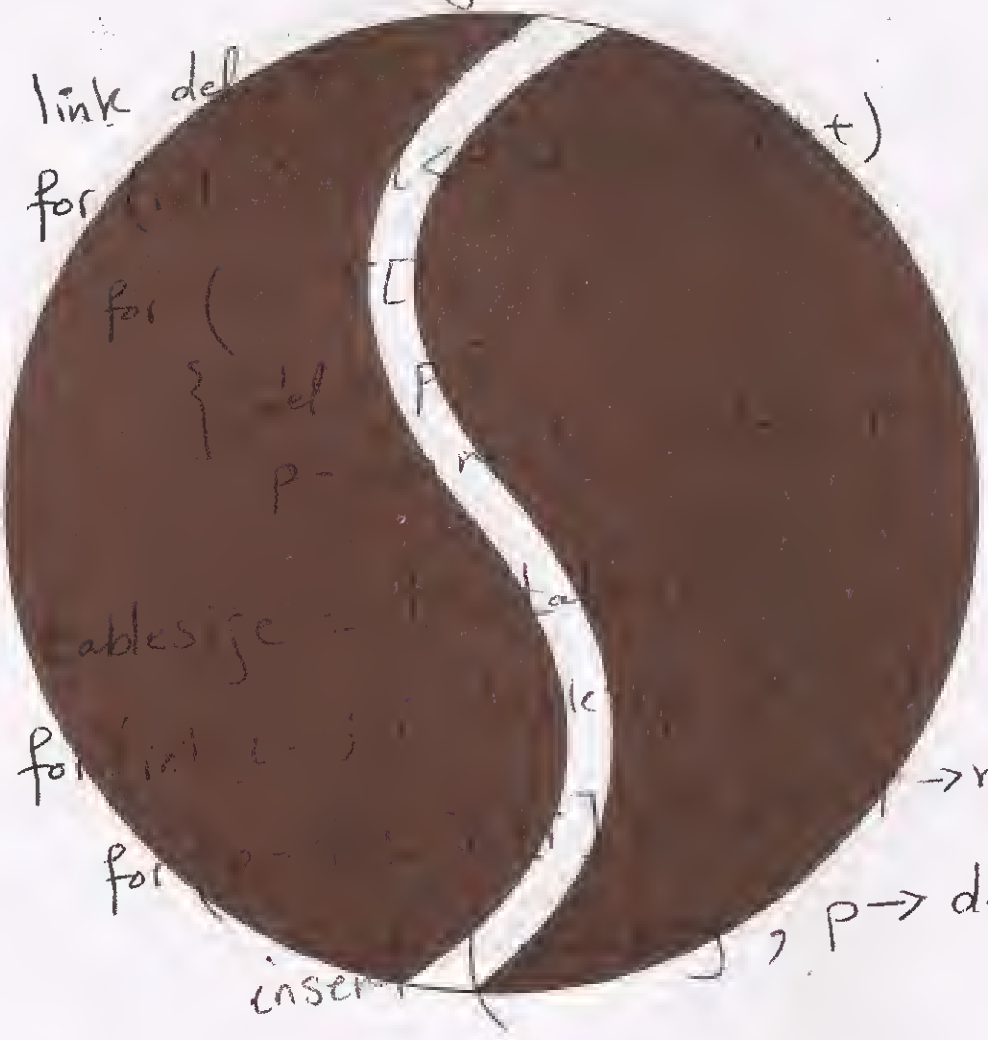
(b) Solved before.

Question 5:

i- solved before.

ii-

```
void Table::Copy(const Table &t1)
{
    link del
    for (
    for (
    {
        del
        p-
    ablesize
    for (int
    for
    insert
}
```



```
                                    t)
                                    )
                                    }
                                    ->next)
                                    , p-> data);
```

Final 2010

Question 1:

```
Const int N=100;
class Polynomial {
private:
    double Arr[N];
    int n;
public:
    Polynomial(
    Polynom
    Polynomial
    Pol
};
```

```
Polynomial::Polynomial
    (unsigned int deg)
{
    assert (deg<N);
    n=deg;
    for(int i=0; i<=n; i++)
        Arr[i]=0;
}
```

```
Polynomial::Polynomial
    (unsigned int deg,
     double a[])
    : (deg<N);
    i<=n; i++)
    [i];

::Add
 & B)
, i
B.n; nmin=n;}
nmx=n; nmin=B.n;}
Polynomial* p=new Polynomial
                (nmx);
for(  i=0; i<=nmin; i++)
p->Arr[i]=Arr[i]+B.Arr[i];

if (n==nmx)
  for(i=nmin+1; i<=n; i++)
    p->Arr[i]=Arr[i];
```

```
else
   for (i=nmin+1; i<=B.n; i++)
      p->Arr[i] = B.Arr[i];

return p;
}
```

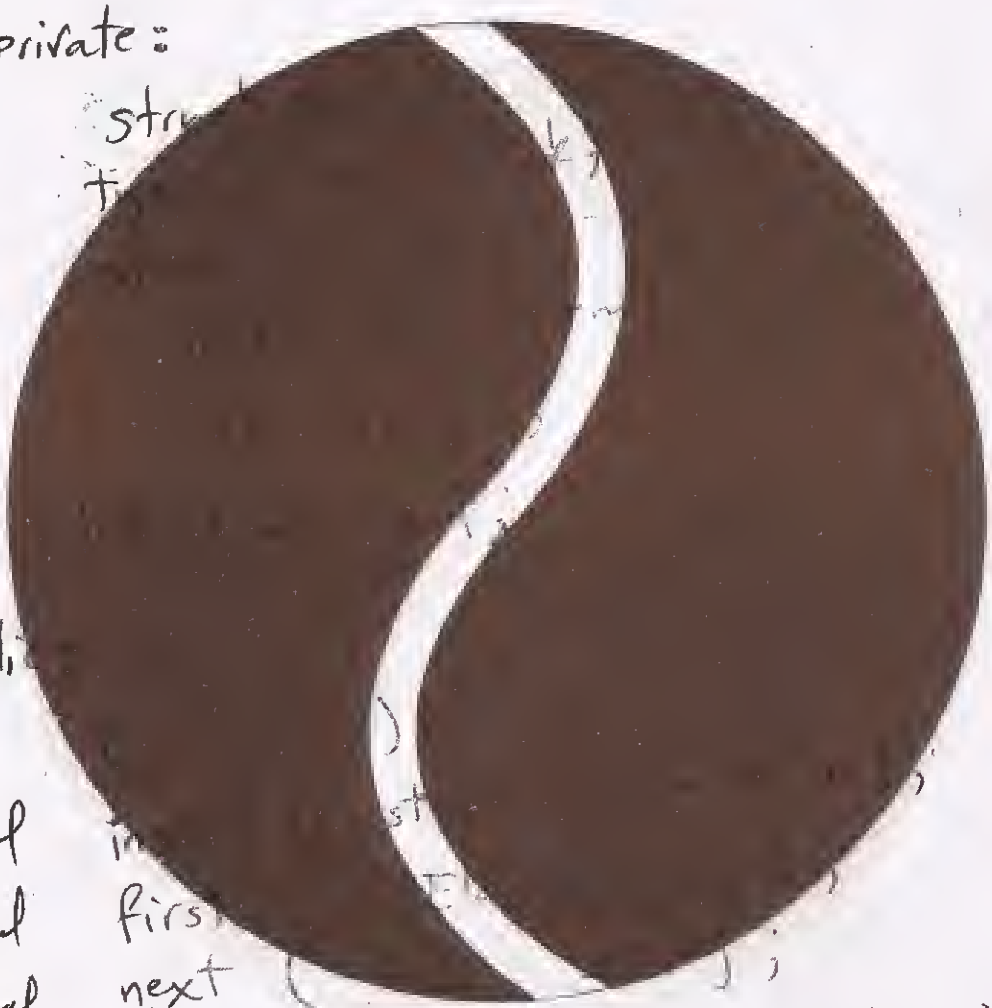
Another solution

```
Polynomial
{
   Polynomial *P;
   if (
   {  p = new
      for
         if (
            p->
         else p->Arr[i] =
   }
   else { p = new Polynomial (B.n);
          for (i=0; i<=B.n; i++)
             if (i<=n)
                p->Arr[i] = Arr[i] + B.Arr[i];
             else
                p->Arr[i] = B.Arr[i];
   }
return p; }
```

Question 2:

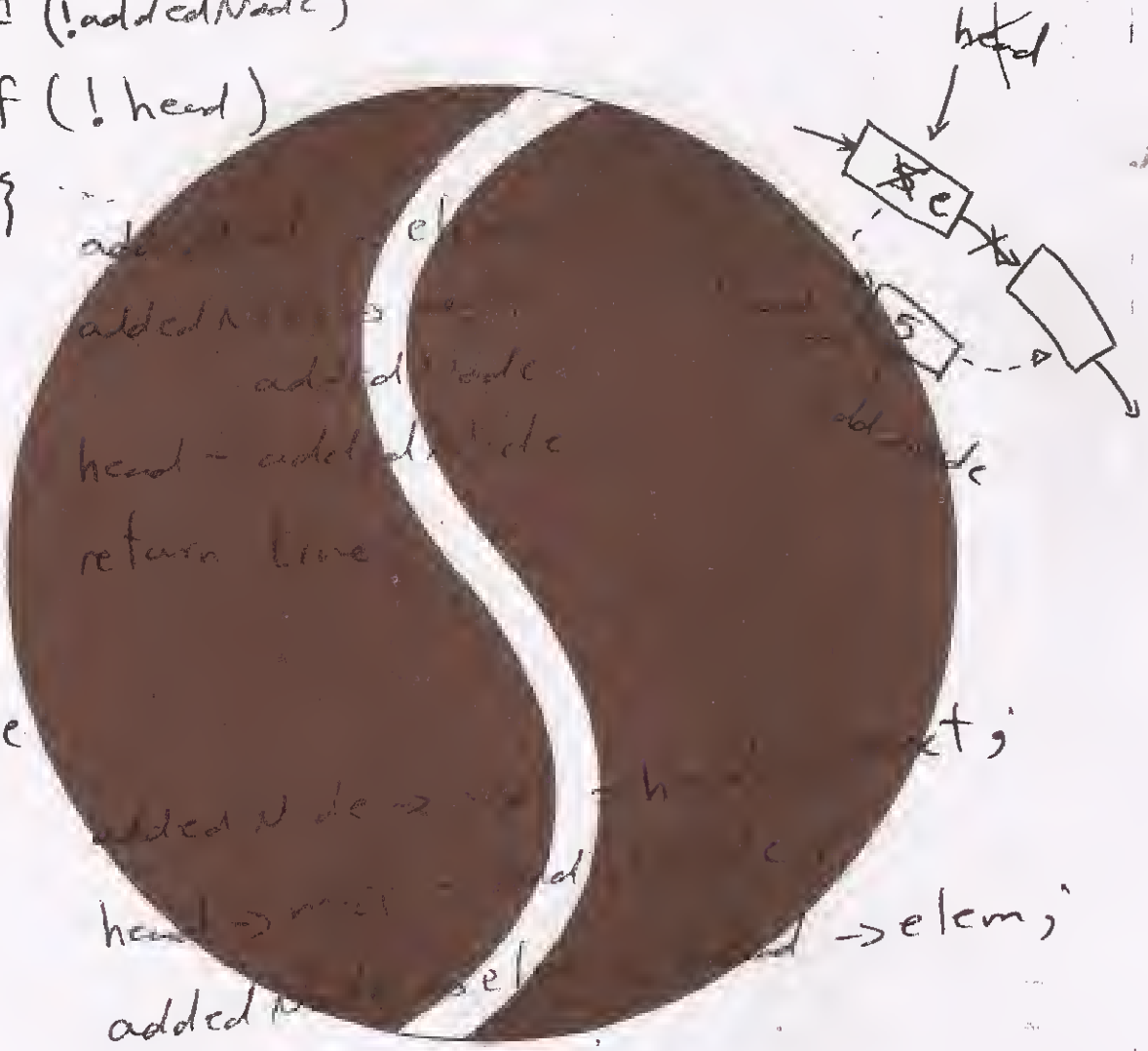
→ Definition:

```
typedef int listElemType;
class CircList {
  private:
    struct
    t
  publi
    bool i
    bool firs
    bool next (
    bool
    void Merge (const CircList &cList);
    void ReversePrint ( );
};
```



→ Functions :

```
bool CircList::insert (const listElemType &e)
{
    link addedNode = new node;
    if (!addedNode) return false;
    if (!head)
    {
        addedNode->elem = e;
        addedNode->next = addedNode;
        head = addedNode;
        return true;
    }
else
    {
        addedNode->next = head->next;
        head->next = addedNode;
        addedNode->elem = head->elem;
        head->elem = e;
        head = addedNode;
        return true;
    }
}
```



- Void CircList::Merge (const CircList &clist)

{ if(clist.head==0) return ;

for (link p= clist.head ; p->next != clist.head ;
p= p->next)

if(! in

or multiple

// Suppo
// node

}

- Void

{ if (! h

link p;

for (p= head ; p->next != head ; p=p->next)

N++;

listElemType * A = new listElemType [N];

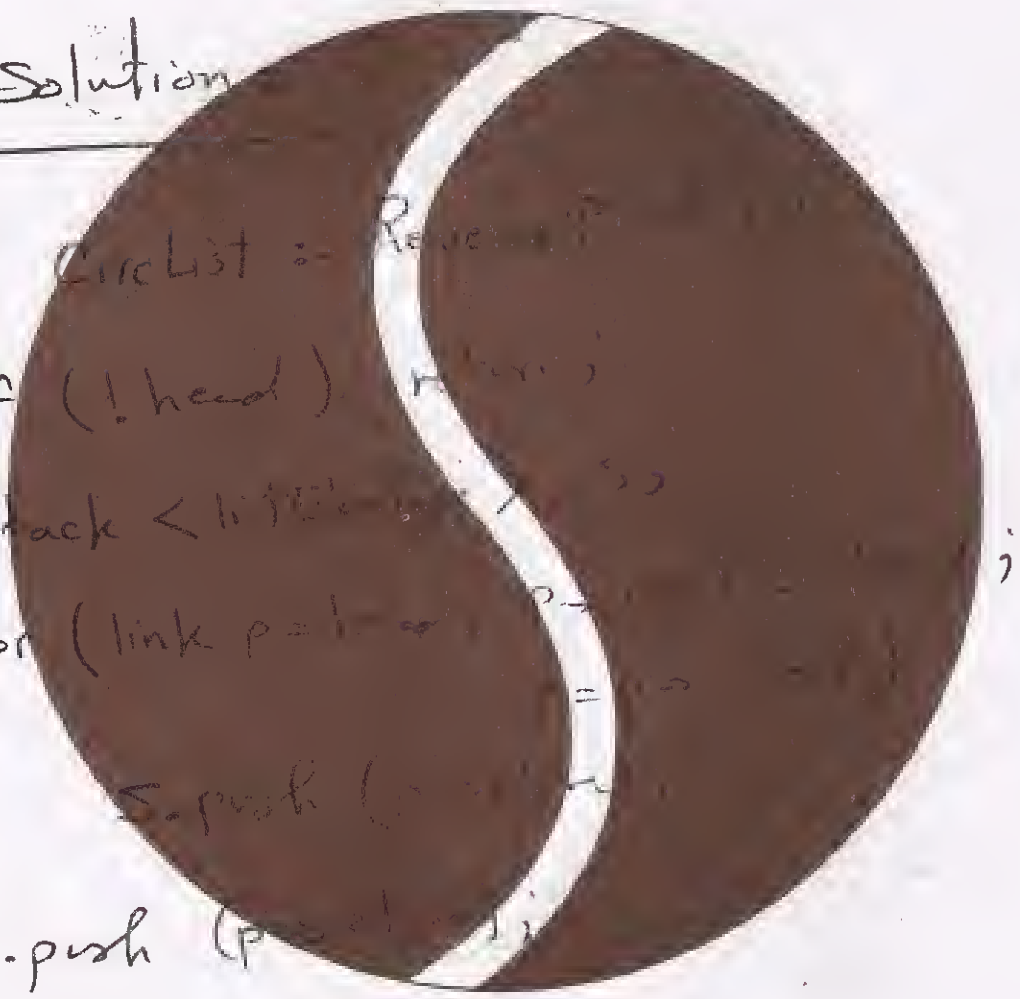
for (p= head ; p->next != head ; p=p->next)

{A[i] = p-> elem; i++;}

A[i]= p->elem;

```
for (i = N-1; i >= 0; i--)
    cout << A[i];
delete [] A;
}
```

Another Solution

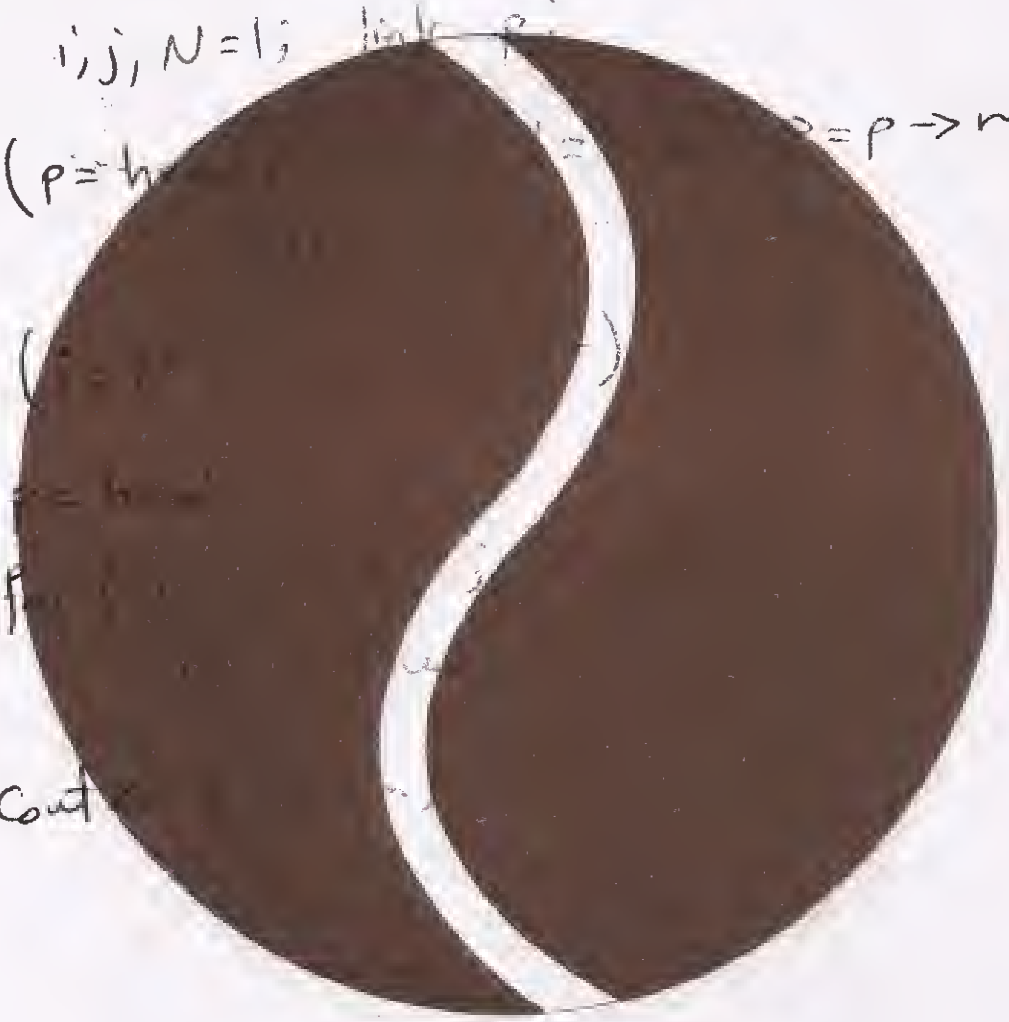


```
void CircList :: Reverse()
{ if (!head)
    stack <li
    for (link p=h
        S.push (
    S.push (p
while (!S.isEmpty())
        cout << S.pop();
}
```

Third Solution :

```
Void CircList :: ReversePrint ( )
{  if (!head) return;
   int i, j, N=1;
   for (p=h                    =p->next)

   for (
   {
      f
      Cout
   }
}
```

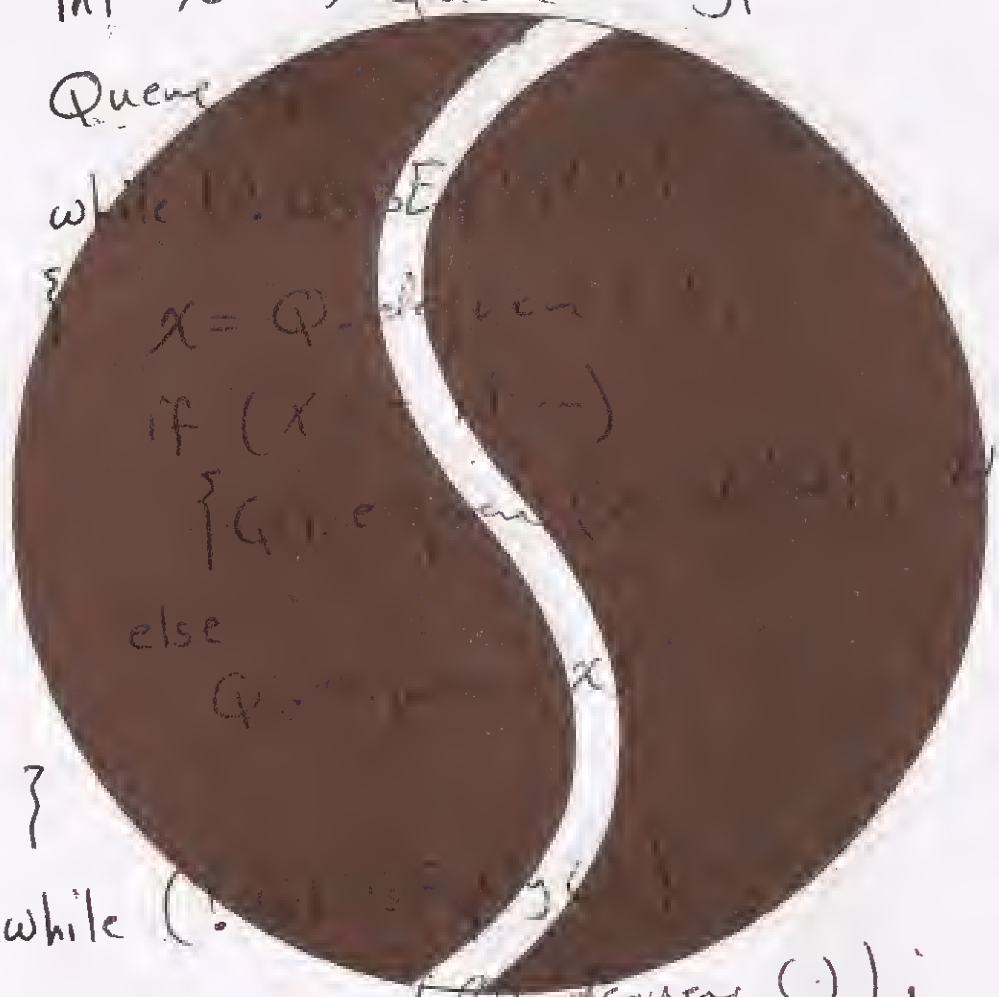


Question 3:

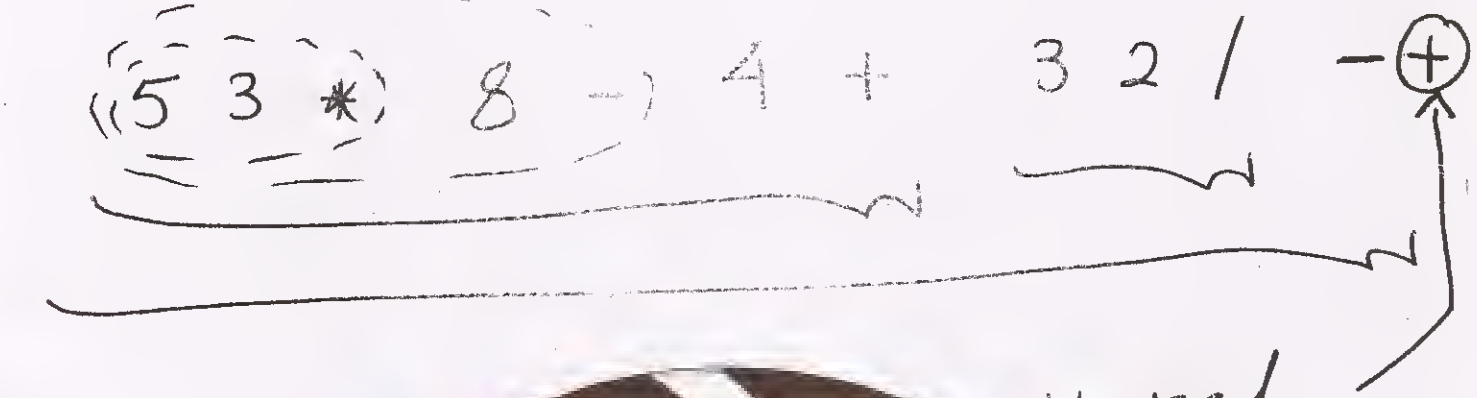
(a)

```
int Replace (Queue &Q, QueueElemType item,
             QueueElemType newValue)
{ int N=0; QueueElemType x;
  Queue
  while
  {
    x = Q.
    if (x
      {Q.e
  else
    Q.
}
while (
  Q.enqueue (Q.dequeue());

return N;
}
```



b) RPN using stacks.



Unused operator

→ Synt

→ ext

→ th

Question 5:

a]

i- T, proof of binary search "Concealed"

ii- F, only in "Concealed"

iii- T

iv- T

v- T

b]

i- Line- Pseudo- F-1 isc = 20

key		bin	Comp.
66	6	6	1
47	7	7	1
87	7	8	2
90	10	10	1
126	6	9	4
140	0	0	1
145	5	5	1
153	13	13	1
177	17	17	1

[39]

285	5	11	7
393	13	14	2
395	15	15	1
467	7	12	6
566	6	16	11
620	0		2
735	15		4

mod 20

→ Pro

→ Fin

2	
3	
4	
5	14
6	66
7	47
8	87
9	126
10	90
11	285
12	467
13	153
14	393
15	395

6
7
735
—

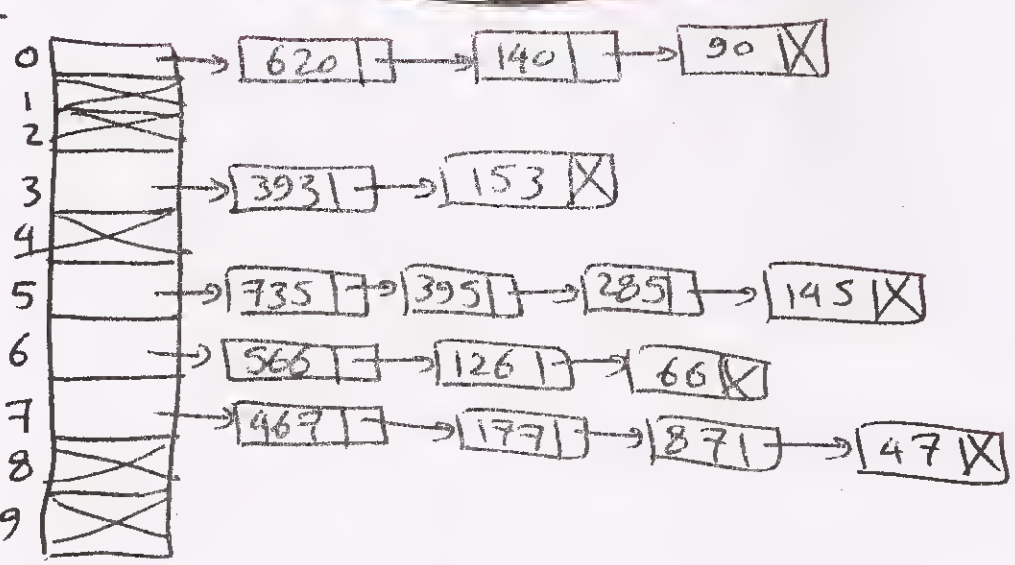
no. of Comp.

87	2
566	11
153	1

ii- Chained Table: Table size = 10

Key	hash o/p	no. of Comp
66	6	3
47	7	4
87	7	3
90	0	3
126	6	2
140	0	2
145	5	4
153	3	
177	7	
285	5	
393	3	
395	5	
467	7	
566	6	
620	0	
735	5	

Table:



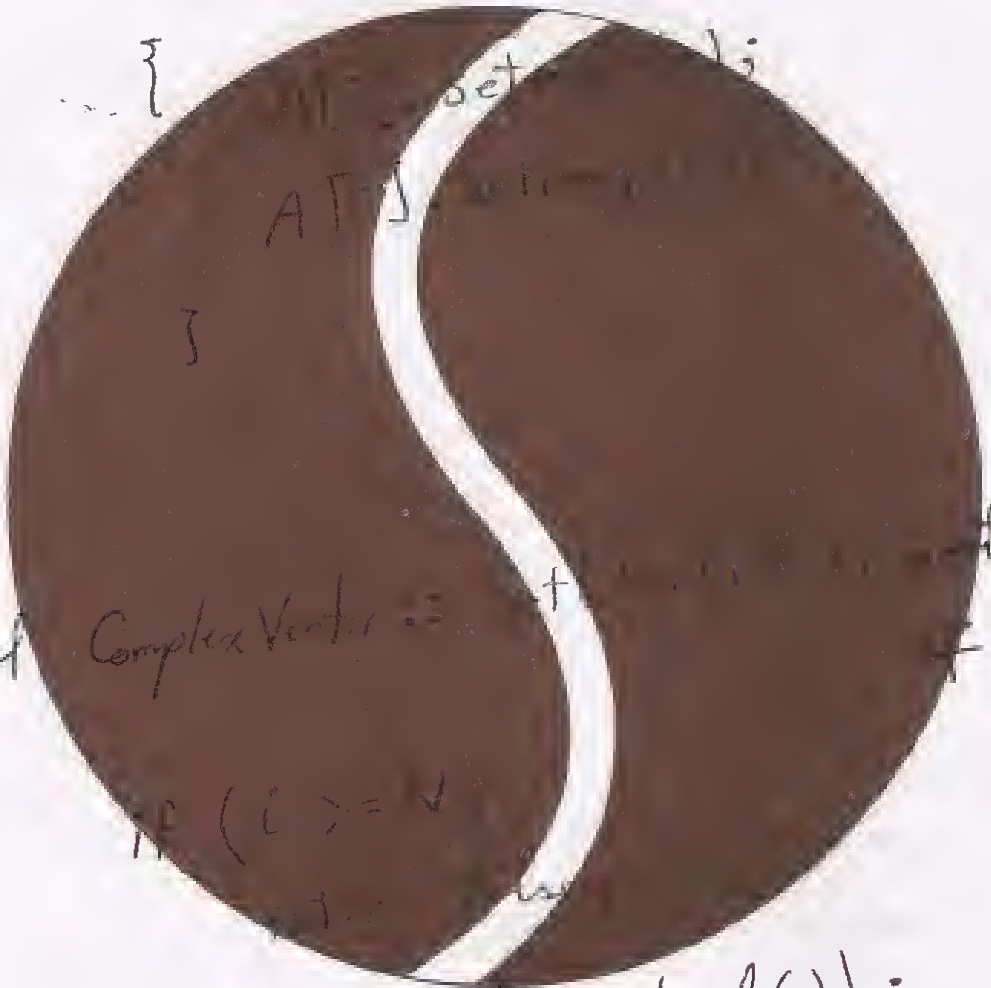
Final 2011

Q1. b.

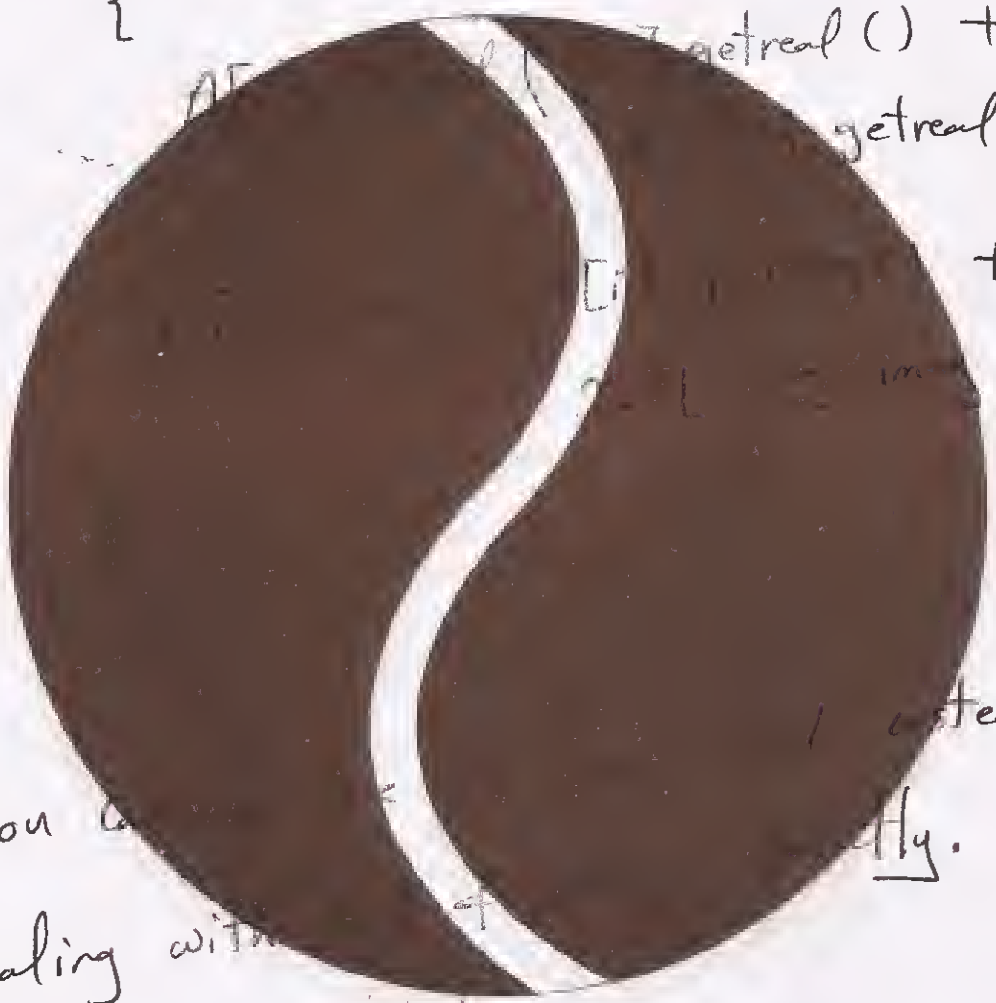
```
#include "Complex.h"
const int
cl
                                    ements
p
                                    );
bool SetElem (int , const Complex &c);
void Add (const ComplexVector &x);
Complex DotProduct (const ComplexVector &x);
};
```

```
ComplexVector :: ComplexVector (int size)
{
    assert (size <= maxsize);
    N = size;
    for (int i=0; i< maxsize; i++)
    {
        A[i] = 0;
    }
}

bool ComplexVector :: operator == (const ComplexVector & c)
{
    if (N != c.N)
        return false;
    for (int i=0; i<N; i++)
        if (A[i] != c.A[i])
            return false;
    return true;
}
```



```
void ComplexVector :: Add (const ComplexVector &x)
{
    assert (N == x.N);
    for (int i=0; i<N; i++)
    {
        A[i] = A[i].getreal() + x.A[i].getreal();
        A[i] = A[i].getimag() + x.A[i].getimag();
    }
}
```



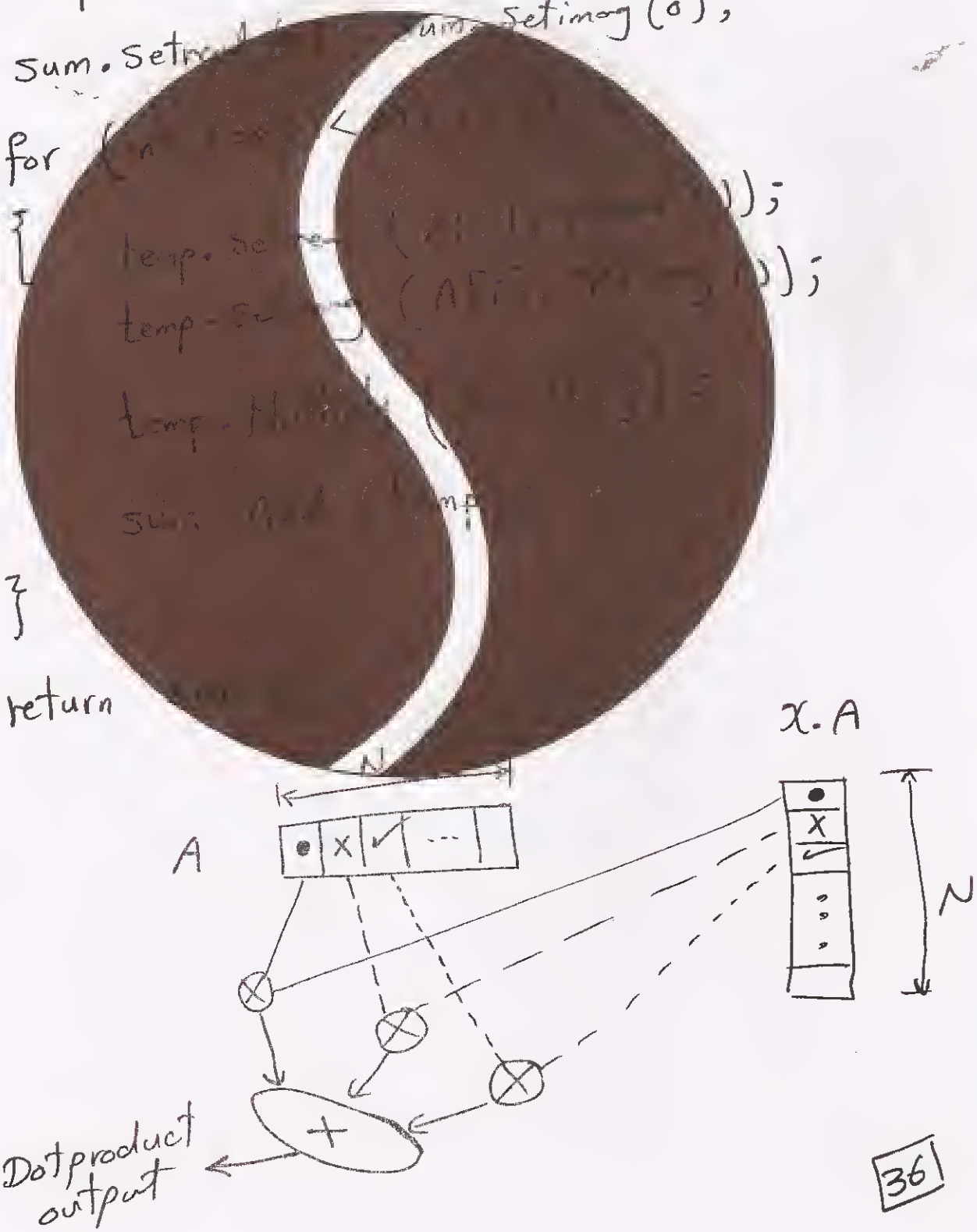
⟹ You can instead of

dealing with ... efficiently.

So, the loop will be :

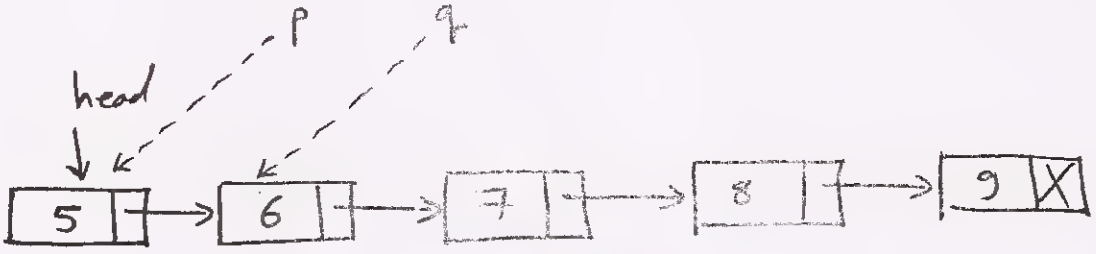
```
for (int i=0; i<N; i++)
    A[i].Add (x.A[i]);
```

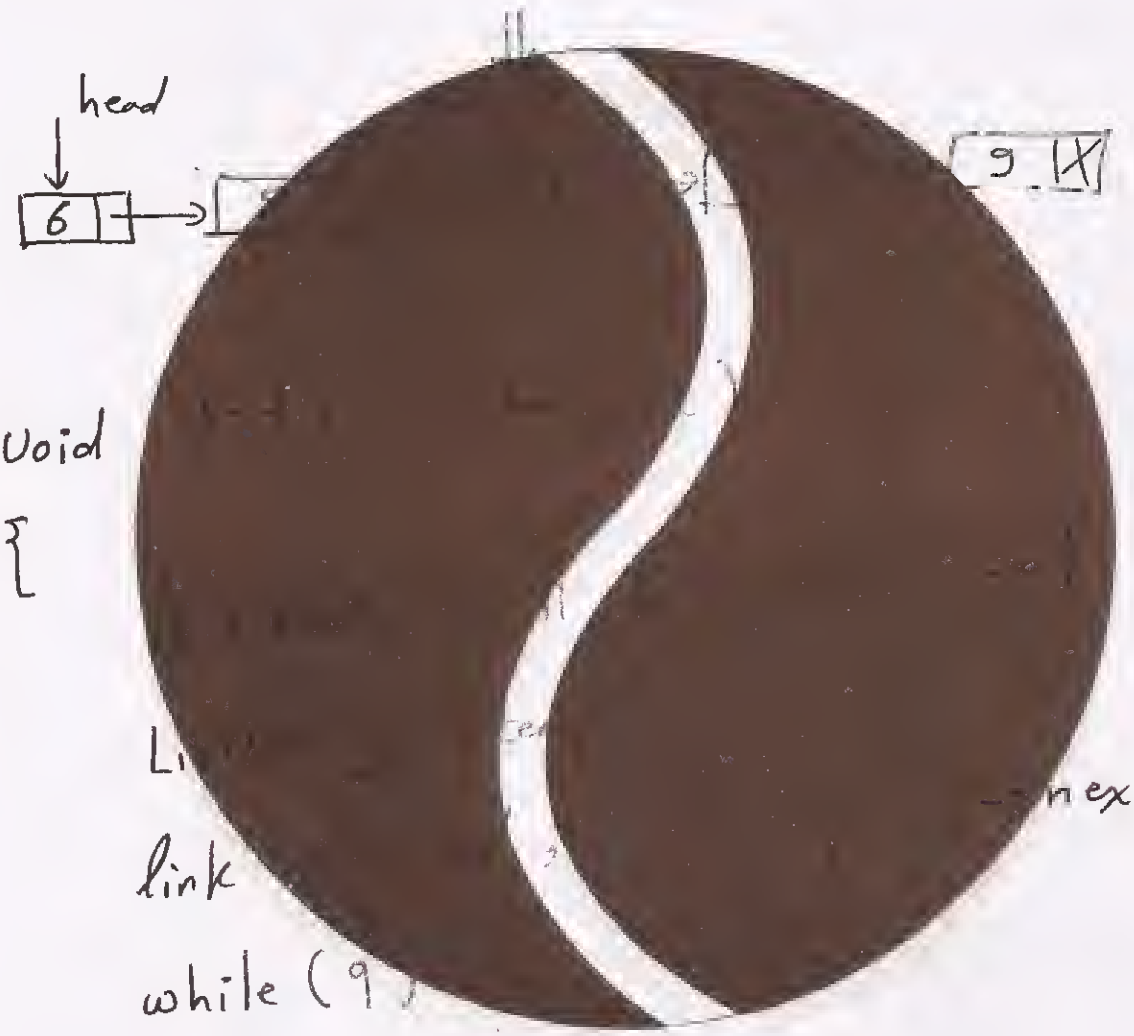
```
Complex ComplexVector :: DotProduct (const ComplexVector
                                     &x)
{
    assert (N == x.N);
    Complex sum, temp;
    sum.Set
    for (n
    {
        temp.Se
        temp.Se
        temp.Mul
        sum.Add
    }
    return
}
```



Q2.

9

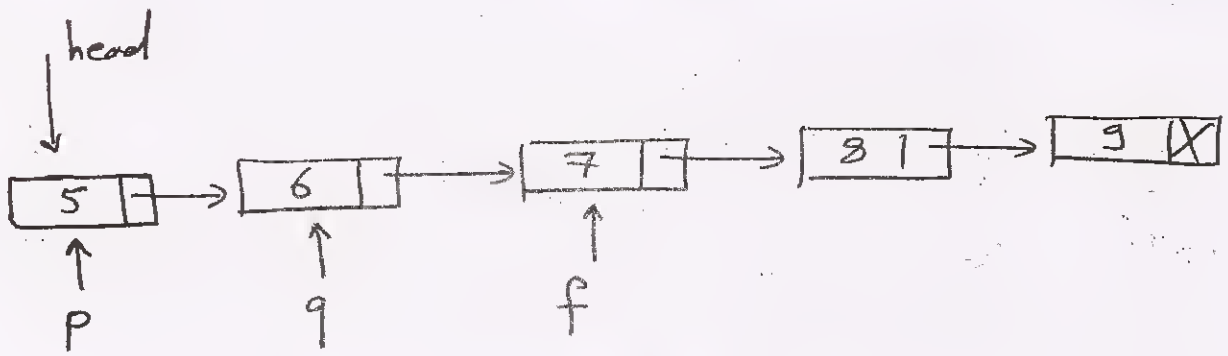





```
void
{
                                        return ;
Li                                  -next;
link
while (9)
{ temp = p-> elem;
  p-> elem = q-> elem;
  q-> elem = temp;

  p = q-> next;
  if (p && p->next) q = p-> next;
  else   q = 0;  }
}
```

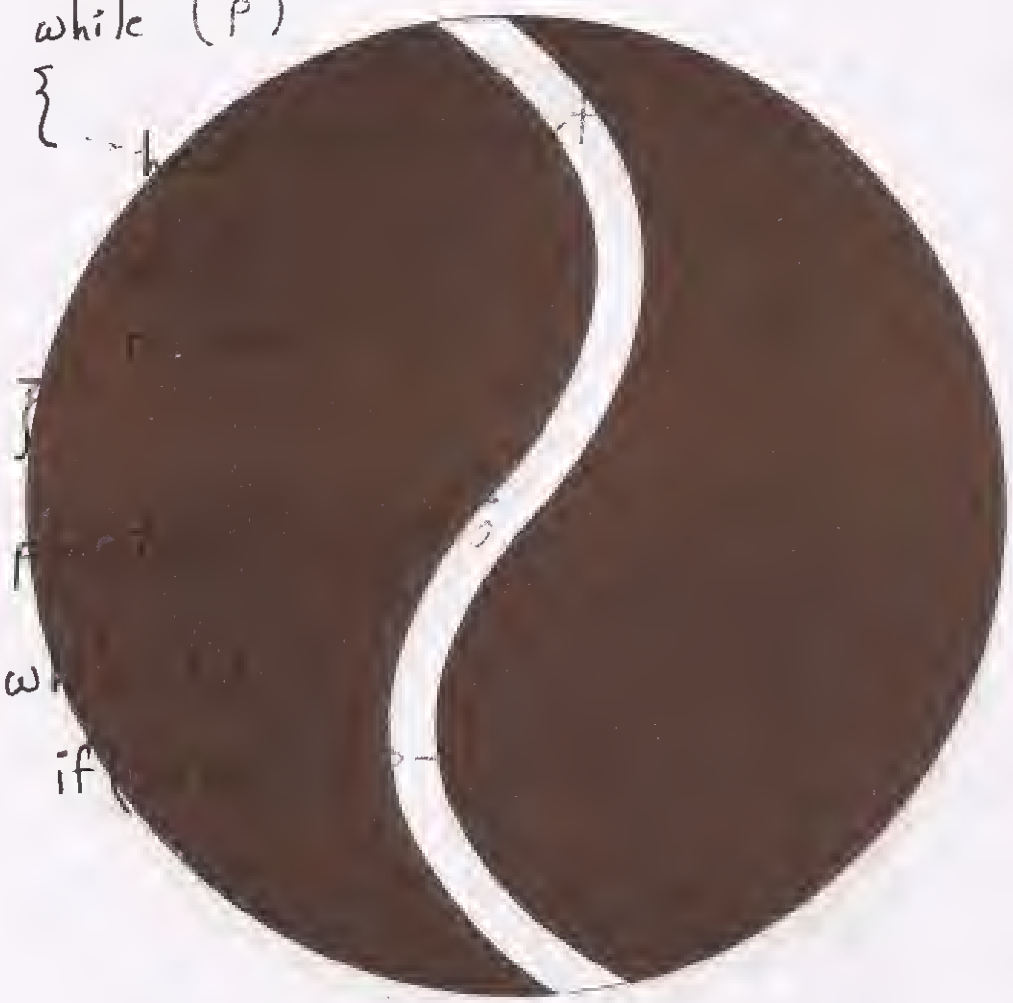
b




```
void list :: Reverse ( )
{
   if (                                  =0 ) return ;
                                              link f;
   link p=
   if ( q->next )
        f= q->
   else f= ;
   p->next =0;
   while ( )
   {
                                  ;
      p =
      q = f;
      if (q)
         f= q->next;
   }
   head =p;
}
```

c

```
void list::copy (const list &otherlist)
{
    link p = head;
    while (p)
    {
```

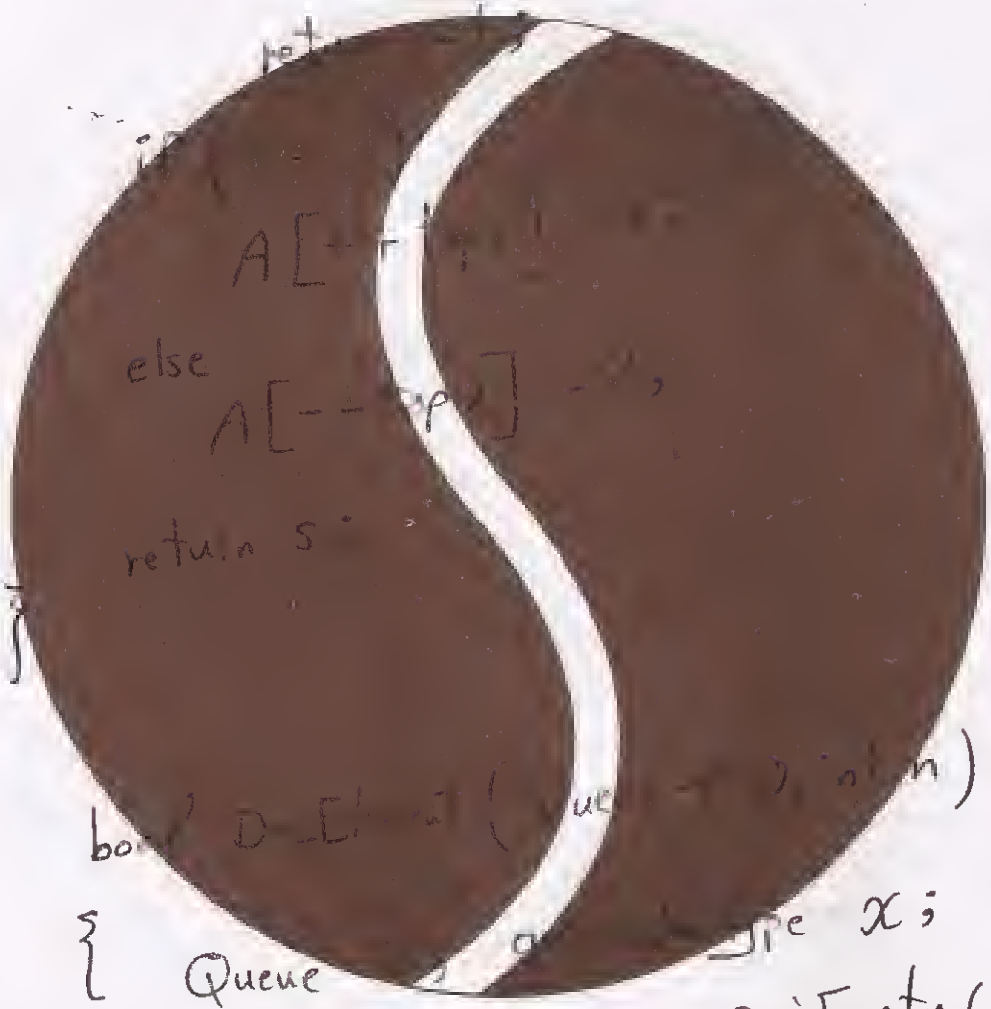


```
}
```

Q3.

a]

```
int push (stackDataType v, int s)
{
    if ( top2 == top1 + 1)
```



```
else
    A[--top2] = v;
return s;
}
```

b]

```
bool D_Empty ( Queue T, int n)
{  Queue Q; DataType x;
   while (n >= 0 && !Q.isEmpty())
      { x = Q.dequeue();
        if (n != 0) T.enqueue (x);
        n--; }
   while (!T.isEmpty()) Q.enqueue (T.dequeue())
   return (n == -1);
}
```

A General Test

Question 1:

Define an abstract data type, the class `ComplexMatrix`, that represents a square matrix of Complex numbers. Each element of the matrix is an object of the `Complex` ADT. The `Complex` ADT supports the operation `Add`, `Multiply`, and the `Set/Get` functions. The `ComplexMatrix` class contains upto 100 complex numbers per dimension and supports the member functions shown in the table below. Write the definition of the `ComplexMatrix` class and implement its member functions. You need to select the proper return [illegible]. You don't need to write the `Complex` class ADT.

i) ComplexMa[illegible] dimn[illegible]	[illegible] to [illegible] f the matrix [illegible] un should [illegible] 0).
ii) SetEle[illegible]	[illegible] ple[illegible] sert[illegible] cu[illegible] e c[illegible] s (i,j). ou s[illegible]uld n firs[illegible]
iii) Mu[illegible]	

Question [illegible]

i. T[illegible] Ad th[illegible] b[illegible]

ii. W[illegible] the[illegible] ts th[illegible] h th[illegible] l ( ) po[illegible] s a[illegible] t mai[illegible] . N[illegible] ists.

Question 3:

i. The follo[illegible] ort [illegible] nd it is called quick[illegible] pre[illegible] ast elements in the array.

```
int partitioning (int a[], int first,[illegible]
{
        int pos=first;
        for (int i=first+1;i<=last;i++)
                if (a[fisrt]>a[i])
                {
                        pos++;
                        swap(a,i,pos);
                }// end of loop
        swap(a,first,pos);
        retrun pos;
}// end of partitioning
```

```
[illegible],int pos1,int pos2)
        int temp=a[pos1];
        a[pos1]=a[pos2];a[pos2]=temp;

void sort (int a[],int first,int last)
{
        if (first>=last) return;
        int pivot=partitioning(a,first,last);
        sort(a,pivot+1,last);
        sort(a,first,pivot-1);
}
```

a) Trace the above code for the following array 12 5 20 -3 9
b) What is the number of comparisons in the worst case of this algorithm?

ii. Write a class BinarySearch to search for certain target in an array of strings recursively.

Question 4:

i. Write a function deloccurrence (stack & s, stackelemtype v) to delete all occurrence of the element v from the stack s

ii. Write a function printprime, to print all prime numbers stored in a given Queue Q.

iii. Write the definition of the standard circular queue class using linked lists and implement the enqueue funct

Question 5:

i. Write the cod nctio urrent, and UpdateCur t to the clas tasks respectiv , add a node at e fron has by the current nter, and update th dat curre

ii. Usin e linked list class def ed ve. Define im p e clas that performs ir sert, lool p, d delete key funct rep sented by an array of li

Test solution

Question 1:

```
#include "complex.h"
const int MAXSIZE = 100;
Class ComplexMatrix {
public:
        ComplexMatrix(unsigned int dimn);
        void SetElem(int i,int j, const
        void Multiply(const Com
private:
//the array for storing elem
        Complex Eleme
        int dim; //the a
}

ComplexMatrix::
{
int i,j;
assert(dimn<MA
dim = dimn ;
for(i=0; i < MA
        for(j=0; j
        {
                Ele
                Ele
        }
}

void ComplexMatrix::Set
Complex &c)
{
        assert(i < dim && j < dim);
        Elements[i][j].SetReal(c.GetReal());
        Elements[i][j].SetImag(c.GetImag());
}
```

```
ComplexMatrix ComplexMatrix::Multiply(const
                        ComplexMatrix &B)
{
        assert(dim==B.dim); int i,j,k;
        lexMatrix Result(dim);
        emp;
        i++)
        <dim;j++)
        =0;k<dim;k++)

        B.Elements[j][k]);
        lt.
        ult
        ld(i
        tur
```

Question 5:

i- **The code of the list will be:**

```
#include<assert.h>
template <class ListElementType>
class List
{
public:
List();
void insert(const ListElementType &
bool first(ListElementType &
bool next(ListElementTy
void UpdateCurrent(
void InsertAtBegin
void deleteCurre
private:
struct Node;
typedef N
struct Nod
{
ListElem
Link next
};
Link hea
Link tail;
Link curre
};
template <c
List<ListEle
{
head = 0;
tail = 0;
current = 0;
}
/////first(), next(), and insert(
ListElementType> ] before
each function and the class name used is [ List<ListElementType> ] like:
template <class ListElementTypc>
void List<ListElementType>::insert(const ListElementType & elem)
//// member functions that we add:
template <class ListElementType>
void List<ListElementType>::UpdateCurrent(const ListElementType & e)
{
assert(current);
current->elem=e;
```

```
}
template <class ListElementType>
void List<ListElementType>::InsertAtBeginning(const ListElementType & e)
{
Link addedNode=new Node;
addedNode->next=head;
addedNode->elem=e;
head=addedNode;
}
template <class ListElementTy
void List<ListElementTy
{
assert(current);
if(head==0)retur
Link del; // if c
if(head==cur
{
del=curren
head=cur
current=
delete de
return;
}
// curren
for(Link
del=curre
p->next=c
current=cur
delete del;
}
```

ii- The cod

```
#include"list.h" //(not
const int MAX_TABLE =
template < class tableKeyTyp
class Table
{
public:
// Table(); we don't need a constructor, since the constructor of list class will cause each
head, tail and
// current to be=0
void insert(const tableKeyType & key, const tableDataType & data);
bool lookup(const tableKeyType & key, tableDataType & data);
void deleteKey(const tableKeyType & key);
void dump();
private:
```

```
struct Slot
{
tableKeyType key;
tableDataType data;
};
List<Slot> T[MAX_TABLE]; // array of linked lists
int hash(const tableKeyType & key);
};
template <class tableKeyType, class tableDataType>
int Table<tableKeyType,tableDataType>::hash(const tableKeyType & key)
{
return key % MAX_TABLE;
}
template < class table
void Table<table
tableDataType
{
int pos(hash
Slot s,s1;
s1.key=ke
bool f=T[
if(f==fals
else
{
while(f)
{ // if the
if(s.key==
f=T[pos].n
}
// if key doe
T[pos]. Insert
}
}


template < class tableKeyT
bool Table<tableKeyType,table
tableDataType & data)
{
int pos(hash(key));
Slot s; bool b;
b=T[pos].first(s);
while(b)
{
if(s.key==key)
{data=s.data;return true;}
```

```
b=T[pos].next(s);
}
return false;
}
```

In the previous function, we hash the key, traverse the list comparing the keys. Remember that s is passed by reference to first () and next() so after the call they contain an element (a slot) from the list.

```
template < class tableKeyType, class tableDataType >
void Table < tableKeyType, tab                    tableKeyType & key)
{
int pos(hash(key));
Slot s; bool b;
b=T[pos].first(s);
while(b)
{
if(s.key==k
{T[pos].de
b=T[pos].
}
}
// delete
delete th
which 'c
template
void Tabl
{
Slot s;bool
for (int i=0;
{
cout << i << '\t';
b=T[i].first(s);
while(b)
{
cout<<s.key<<'\t';
b=T[i].next(s);
}
cout << '\n';
}
cout << '\n';
}
```

Question 2:a

```
void list:: Sort ()
{
   if(head ==0 || head-> next ==0)
                return;
   link t =head;
   while (t->
          t=
   link p2 = t;
   while (t->p
   {
      while (p1
      {  if( p1->el
      } t = p
        p1
        p1
      }
      p2=p1;
      p1 = p1->prev;
   }
   t= t->prev;  p2=t;
   if (p2->prev)   p1 = p2->prev;
}
}
```